REMARKS

No claims have been amended, cancelled or added. Hence, claims 1 – 24 are pending in the Application.

SUMMARY OF REJECTIONS/OBJECTIONS

Claims 1 – 24 are rejected under 35 USC 102(e) as being anticipated by U.S. Patent No. 6,438,562, herein *Gupta*.

CLAIMS 1 AND 13

Claims 1 and 13 recite:

constructing work granules that manipulate rows in a manner that groups the rows within said work granules according to logical storage units that contain the rows; and

during execution by an entity of a particular work granule that involves manipulation operations for rows in a particular logical storage unit:

causing said entity to perform said manipulation operations for rows completely contained in said logical storage unit;

determining a set of spanning rows that are partially contained in said logical storage unit and that satisfy a particular condition; and

causing said entity to perform said manipulation operations for all pieces of all spanning rows in said set of spanning rows.

The system of claims 1 and 13 recite a novel and advantageous approach for modifying data in spanning rows. A spanning row is a row that is stored in more than one logical storage unit, such as a data block. The portions of rows are referred to as row pieces. (See, for example, Application, page 3, lines 8 – 10). The approach claimed requires that an entity, such as a slave process, perform manipulation operations for all pieces of a spanning row when the spanning row satisfies a condition. An example of such a condition is that a spanning row start in a logical storage unit (see claim 2).

The Office Action has alleged that *Gupta* anticipates claims 1 and 13. In order for a reference to anticipate a claim that describes steps related to processing spanning rows, the reference would have to disclose or at least suggest the concept of a spanning row. *Gupta* does not do this.

The Office Action appears to allege that parts of *Gupta* disclose spanning rows. One of these parts is the depiction of items 211 – 216 in FIG. 2 (See Office Action, page 3, 1st paragraph). *Gupta* describes these items as rows in tables with columns, as follows.

> Relational databases store information in indexed tables that are organized into rows and columns. FIG. 2 illustrates a sample table for an example relational database. **Each row as 210 in** the table 200 represents an individual record. Each column 220 in the table represents a different kind of information or "attribute" that is of interest for all the records. For example, Table Emp 200 stores employee records which contain columns 220 that correspond to the following attributes: employee name (Empname), employee number (Empno), social security number (SSN), department number (Deptno), Salary, job title (Title), date of employment (Startday), etc., in columns 221, 222, 223, 224, 225, 226 and 227, respectively. Each row 210 in the Table Emp 200 stores the same attributes for each individual employee, one attribute per column 220, but the values of an attribute stored in a column 220 may change. In this example, an employee record is uniquely identified by a social security number, SSN, column 223, or an employee number, Empno, column 222. In tables where none of the attributes are unique, a unique attribute called a ROWID may be generated for the record by the database. (col. 2, lines 20 – 41)

Nothing in the above excerpt or elsewhere in *Gupta* implies or suggests that any portion of any row from rows 210 is stored in more than one logical storage unit, as is required for the spanning row claimed.

The following is another section that the Office Action seems to allege discloses spanning rows. (Office Action, page 2, last paragraph)

> When the CP distributes the data manipulation work granules, the number of PDML slaves utilized may vary. For example, if the PDML operations are partitioned the same way as the database table, then only three PDML slaves will be used, those on node 111, 112 and 113, respectively, because these are the nodes with partitions of the table. Node 114 is not used. Alternatively, to make the most use of available processors, some of the PDML operations may be assigned to node 114. Because node 114 has less affinity for the rows of the database tables than the other three nodes, node 114 may be assigned a work granule with fewer than one fourth the total DML operations, according to various schemes for distributing tasks known in the art.
> As mentioned above, the number of slaves used during the data manipulation phase of a PDML operation need not equal the number of slaves used during the index update phase of the PDML operation. Thus, even if the DML operations are confined to nodes 111, 112 and 113, all four nodes may still be utilized for the index update slaves. Only node 112 has a strong affinity for the global index in the example illustrated in FIG. 7. Thus, node 114 is no less qualified to update the global index than are nodes 111 or 113. Consequently, even if only three PDML slaves are used, the CP can choose to divide the index update task among four index update slaves, with index update slave 742 on node 112 perhaps receiving a greater share of the load because of its greater affinity for the disk on which the global index resides, again, allocating shares according to various schemes for distributing tasks known in the art. (col. 14, lines 62 – col. 15, line 24)

The above excerpt clearly describes distributing among a set of slaves the work of performing DML operations upon rows. However, it does not disclose nor suggest in any way that those rows are spanning rows.

The following is another section that has been alleged to disclose spanning rows. This section was cited for claim 8 as disclosing or suggesting to insert a row piece of a spanning row. (Office Action, page 5, first full paragraph)

Data in indexed tables, such as Table Emp 200 in FIG. 2, are manipulated with a set of commands which can be called Data Manipulation Language (DML) commands. Typically, the DML commands supported by database systems include, for example, commands to delete rows, insert rows, and update rows. The update row operation is often implemented as a delete row followed by an insert row. The delete row, insert row, and update row operations are referred to hereinafter as insert, delete, and update.

In a multiple-node system, to make use of the multiple nodes, database servers have been designed to support partitioned tables and parallel processing of DML commands. **In a partitioned table, the rows of the table are grouped into distinct partitions that may be spread over multiple nodes.** For example, FIG. 1 illustrates a database table that is partitioned into three partitions, Partition A 161, Partition B 162 and Partition C 163 on the disk banks 151, 152 and 153, respectively, local to nodes 111, 112, and 113, respectively.

The division of rows into partitions is usually based on the value of a table partition key defined by one or more columns of the table. For example, the Table Emp 200 can be divided among the three disk banks using employee number, Empno, in column 222 as the partition key. For purposes of illustration, Empno values from 1-100 may be stored in Partition A on disks 151 local to node 111, records of employees 101 to 200 may be stored in Partition B on disks 152 local to node 112, and rows for employees 201 to 300 may be stored in Partition C on disks 153 local to node 113.

Each node can maintain an index for the rows in its own partition of the table. Such indexes, which only index the rows that belong to a partition of a table and not the entire table, are referred to as local indexes. At least one of the nodes, e.g. node 113, may maintain a global index 170 for all the rows of the Emp table. A global index is an index that contains index entries for all rows of the table.

Clearly the excerpt above discusses the "division of rows into partitions". However, this section is merely teaching dividing multiple rows into groups of row, each grouping being stored in different partitions. ("In a partitioned table, the rows of the table are grouped into distinct partitions that may be spread over multiple nodes.") Claims 1 and 13, require something else other than dividing multiple rows into groups of rows. Claims 1 and 13 in particular require dividing a single row into logical storage units. A teaching that teaches to divide multiple rows and store them in multiple partitions does not disclose or suggest in any way to divide a single row and store it in multiple logical storage units.

As shown above, the cited art does not disclose or suggest in any way a spanning row, as required for the claims 1 and 13. Therefore, the cited art cannot possibly suggest the claimed steps involving spanning rows. Accordingly, claims 1 and 13 are patentable. Reconsideration and allowance of claims 1 and 13 is respectfully requested.

CLAIMS 8 AND 20

Claims 8 and 20, recite:

inserting a first row piece of a spanning row into a first logical storage unit;

prior to inserting a second row piece of said spanning row into a second logical storage unit, determining whether one or more criteria is satisfied, wherein said one or more criteria include that said second logical storage unit has enough space allocated to identify at least a threshold number of interested transactions; and

inserting said second row piece of said spanning row into said second logical storage unit only when said one or more criteria are satisfied.

Claims 8 and 20, like claims 1 and 13, describe steps that involve spanning rows. Claims 8 and 20, in fact, explicitly cite separate row pieces of the spanning row and inserting the row pieces into separate logical storage units. For reasons similar to those discussed with respect to claims 1 and 13, the cited art fails to disclose spanning rows as claimed, much less inserting separate row pieces of a spanning row into separate logical storage units.

DEPENDANT CLAIMS

The pending claims not discussed so far are dependant claims that depend on an independent claim that is discussed above. Because each of the dependant claims include the limitations of claims upon which they depend, the dependant claims are patentable for at least those reasons the claims upon which the dependant claims depend are patentable. Removal of the rejections with respect to the dependant claims and allowance of the dependant claims is respectfully requested. In addition, the dependent claims introduce additional limitations that independently render them patentable. Due to the fundamental difference already identified, a separate discussion of those limitations is not included at this time.

For the reasons set forth above, Applicant respectfully submits that all pending claims are patentable over the art of record, including the art cited but not applied. Accordingly, allowance of all claims is hereby respectfully solicited.

Respectfully submitted,

HICKMAN PALERMO TRUONG & BECKER LLP

[illegible]

Dated: March 31, 2003

Marcel K. Bingham
Reg. No. 42,327

1600 Willow Street
San Jose, CA 95125
Telephone No.: (408) 414-1080 ext.206
Facsimile No.: (408) 414-1076

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Box Non-Fee Amendment, Commissioner for Patents, Washington, DC 20231.

on March 31, 2003 by Trudy Bagdon
(Date) (Signature)